Playing around with Hidden Markov-chain toy model

Anna Antoniou

May 2020

1 Hidden Markov-Chains: what are they?

Historically, Markov-chains were developed in the early 1970s by Russian mathematician Andrey Andreyevich, and have found application in many areas such as speech recognition and in the analysis of biological sequences. Nowadays they are considered to form a specific part of dynamic Bayesian networks.

A Hidden Markov chain model (HMM) is a statistical model based on the Markov chain. The latter refers to a collection or sequence of random variables (X_t where t denotes time or the order of the variable in the sequence) which obeys a special property. Namely, it assumes that a future state only depend on the current state, and the states before the current one have no impact on the future state. In other words, a Markov chain model assumes any future event is conditionally independent from past events. However, sometimes the event is not directly observable. In a HMM, the system being modelled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters.

Such a model is useful in helping us calculate the probabilities of a sequence of events, by capturing hidden information from the observable sequence. "A good HMM accurately models the real world source of the observed real data and has the ability to simulate the source. A lot of Machine Learning techniques are based on HMMs have been successfully applied to problems including speech recognition, optical character recognition, computational biology and they have become a fundamental tool in bioinformatics: for their robust statistical foundation, conceptual simplicity and malleability, they are adapted fit diverse classification problems.

Here in my effort to introduce myself to HMMs from a practical point of view, I follow an introductory example by Udemy. the simple example of a friend trying to predict the weather in a different city based on her friend's mood who lives there. An HMM then allows you to infer the hidden state (weather) based on another observable measurement (you friend's mood).

Figure 1: Jen is to guess the weather based on Bob's mood. She can only observe Bob's mood, but not the weather.

2 Introductory example

Here we follow an introductory example by Udemy in order to introduce HMM's from a practical point of view. *Word of caution as the video starts going rogue at the 29:00 minutes mark. But not to worry, we account for that!* We also note that all images thereafter belong to Udemy.

The example assumes that Jen, who is business partners with Bob, lives in Paphos[1] and Bob lives in London. Now, Bob's mood is affected by the whether it is Sunny or Rainy. Jen is to guess, judging by Bob's mood, whether London is Sunny or Rainy.

Now Jen knows that Bob is mostly happy when it is Sunny and *mostly* grumpy when it is Rainy. Jen knows that when it is Sunny, Bob is happy with a probability of 0.8 and grumpy with a probability of 0.2, and when it is Rainy Bob is happy with a probability of 0.4 and grumpy with a probability of 0.6. There are coined emission probabilities. In mathematical notation, if we let S, R denote Sunny and Rainy respectively, and H, G denote happy and grumpy respectively, then we say that:

$$\begin{aligned} P(H|S) &= 0.8 \\ P(G|S) &= 1 - P(H|S) = 0.2 \\ P(H|R) &= 0.4 \\ P(G|R) &= 1 - P(H|R) = 0.6 \end{aligned}$$

Jen also knows the probabilities of transitions between Sunny and Rainy weather as well as the probabilities of the weather staying the same from one day to the next (Figure 2). For example, the probability of rain after a sunny day is 0.2 whereas the probability of the sunny weather remaining is 0.8.

[1] Certain creative liberties have been taken.

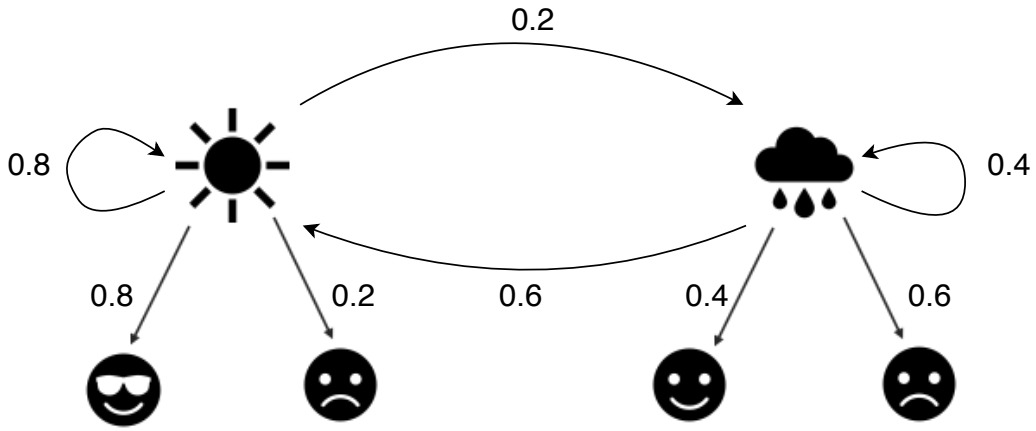

Figure 2: Hidden Markov model. Jen only gets to observe Bob's mood, but not the weather; the weather is a hidden state. The transition probabilities are transition probabilities for the hidden states. Emission probabilities, observations are emitted from the hidden states

3 Questions

3.1 How did Jen find these probabilities?

Data! She collected data, and observed that on 8 occasions a Sunny day is followed by a Sunny day (SunnySunny) and on 2 occasions a Sunny day is followed by a Rainy day (Figure 3). She also observed previously that 8 out of ten times, Bob is happy when Rainy and so on (Figure 4).

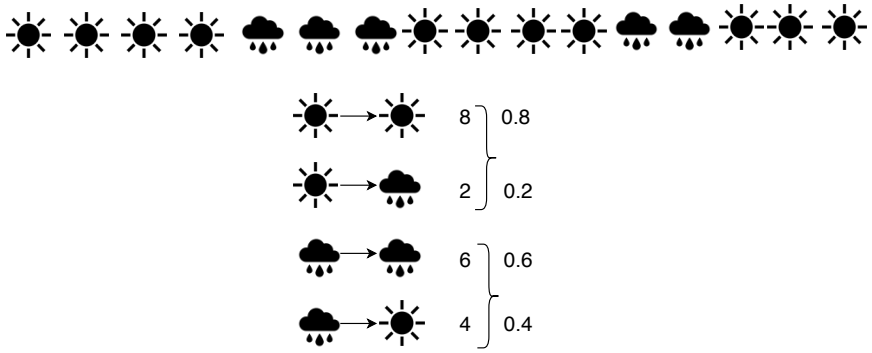

Figure 3: Jen used previously collected data to calculate the transition probabilities.

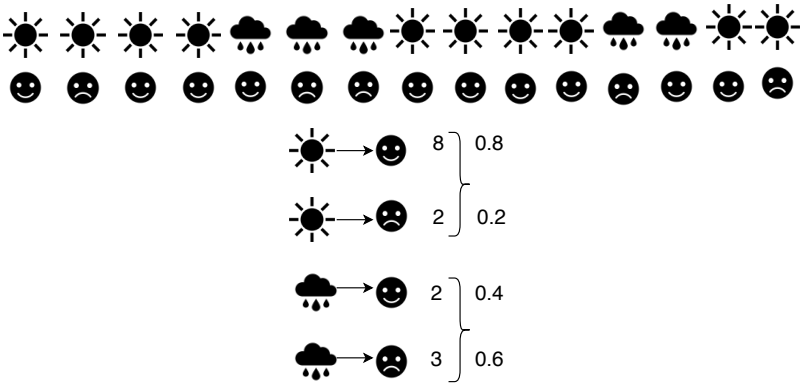

Figure 4: And she also used her observations to come up with the emissions probabilities.

3.2 What is the probability that a random day is Sunny or Rainy?

There is two ways of doing this. Either Jen collected weather data for say 100 days (Figure 5), or she used the transition probabilities to calculate this. For example, if today is Sunny, then yesterday was either Sunny or Rainy, or is today is Rainy, yesterday was either Rainy or Sunny. Together with the law of total probability, we can solve the following set of equations:

$$P(S) = P(S|S) + P(S|R)$$
$$P(R) = P(R|S) + P(R|R)$$
$$1 = P(R) + P(S)$$

leading to $P(S) = 2/3$ and $P(R) = 1/3$.

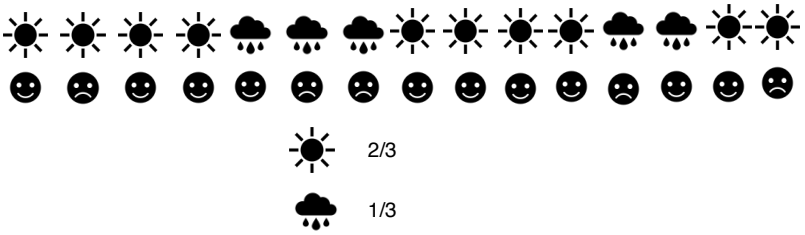

Figure 5: In order to calculate the probability that a random day is Sunny or Rainy, Jen can either use the transition probabilities, or observe the weather for a period of time.

3.3 If Bob is happy today, what is the probability that it is Sunny or Rainy?

In order to address this question we need to utilise Bayes theorem of conditional probabilities. The probability that is it Sunny given Bob is happy then becomes:

$$\begin{aligned} P(S|H) &= \frac{P(S,H)}{P(H)} \\ &= \frac{P(H|S)P(S)}{P(H|S)P(S) + P(H|R)P(R)} \\ &= \frac{0.8 \times 2/3}{0.8 \times 2/3 + 0.4 \times 1/3} \\ &= 0.8 \end{aligned} \tag{1}$$

Equally, the probability that it is Rainy given Bob is happy is equal to $1 - P(S|H)$, or:

$$\begin{aligned} P(R|H) &= \frac{P(R,H)}{P(H)} \\ &= \frac{P(H|R)P(R)}{P(H|S)P(S) + P(H|R)P(R)} \\ &= \frac{0.4 \times 1/3}{0.8 \times 2/3 + 0.4 \times 1/3} \\ &= 0.2 \end{aligned} \tag{2}$$

3.4 If for three days Bob is Happy, Grumpy, Happy, what was the weather?

In order to address this question, we will essentially be using maximum likelihood. Notable, there are in total 2^3 possible combinations to be considered here (Figure 6).

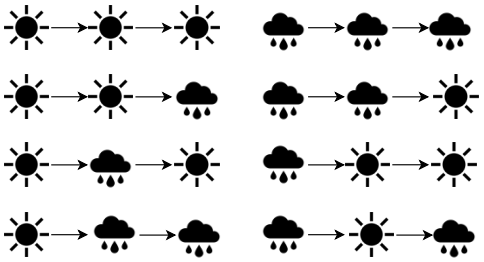

Figure 6: All the possibilities for the 3 days.

Let's start with a simpler question and consider only the first two days. In this simpler scenario, one possibility to consider is that both days were Sunny (Sunny $\rightarrow$ Sunny). Let's find the probability of event that the first day was Sunny (S_1) given Bob was Happy (H_1) and that the second day is

also Sunny (S_2) given Bob was Grumpy (G_2) and that the previous day was Sunny. In equations, and with the help of Bayes' theorem, this boils down to:

$$\begin{aligned} P(S_1, S_2|H_1, G_2) &= \frac{P(H_1, G_2, S_1, S_2)}{P(G_2, H_1)} \\ &= \frac{P(G_2|S_2) \times P(S_2, S_1, H_1)}{P(G_2) \times P(H_1)} \\ &= \frac{P(G_2|S_2) \times P(S_2|S_1, H_1) \times P(S_1, H_1)}{P(G_2) \times P(H_1)} \\ &= \frac{P(G_2|S_2) \times P(S_2|S_1) \times P(H_1|S_1) \times P(S_1)}{P(G_2) \times P(H_1)} \end{aligned} \tag{3}$$

We know all values for the numerator. Now, let's turn our attention at the denominator. The probability that Bob is Grumpy or Happy is the collection/summation of all the marginals, namely:

$$P(G) = P(G|S)P(S) + P(G|R)P(R) \tag{4}$$

and

$$P(H) = P(H|S)P(S) + P(H|R)P(R) \tag{5}$$

Substituting the above back into equation 3, the probability of Sunny → Sunny given Bob was happy then grumpy becomes:

$$\begin{aligned} P(S_1, S_2|H_1, G_2) &= \frac{0.2 \times 0.8 \times 0.8 \times 2/3}{(0.2 \times 2/3 + 0.6 \times 1/3) \times (0.8 \times 2/3 + 0.4 \times 1/3)} \\ &\approx \frac{0.086}{0.332 \times 0.668} \\ &\approx 0.39 \end{aligned} \tag{6}$$

Notice that in the video, the instructor does not care about the actual value of the denominator. This is because it is constant for all weather possibilities. Since we are trying to find the combination of weather that is the most likely, and we will be essentially comparing numerators.

The idea is to solve a similar equation for all the 2^3 combinations, and find the most likely hidden states. You can see how the number of combinations quickly makes this approach highly impractical: for n days, there are 2^n combinations, an exponential growth!

To circumvent laborious hand calculations, the instructor suggests an algorithmic approach: Viterbi's algorithm. The main idea behind this algorithm is to find the most likely path, done at each time-step. This involves a calculation at each time-step of the probability of the given state given every possible state in the previous state.

However, when it comes to picking the most probable path, things could easily go wrong. In order to built the hidden state path, it is a common mistake to select the path with the maximum probability at each day or time step (as seen in the video). This ignores the basic premise of HMMs:

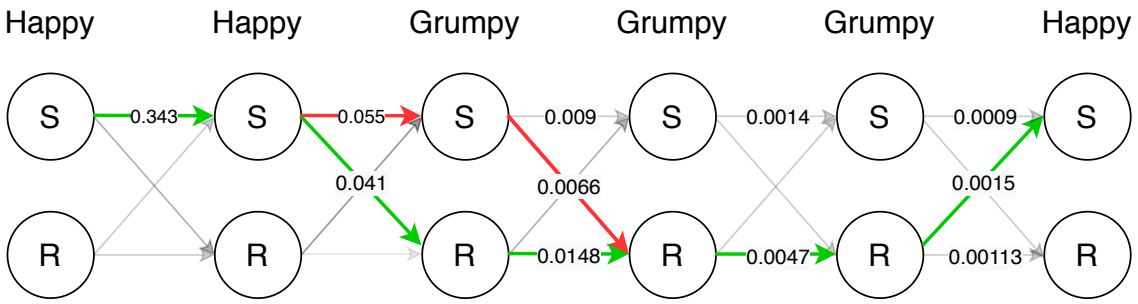

Figure 7: HMM example trellis. When one selects the hidden states based on current maximum probability, one can miss the most probable path. Here, one will select hidden state 'S' at the third step, leading to another path. In addition, in this example, 'R' is chosen at step 4, based solely on 0.0148, which of course comes from the previous state 'R'.

the current state depends solely on the previous state. By doing so, one ignores the previous state that maximises the current state, selecting the wrong path.

To illustrate this, let's built a trellis to represent the problem (Figure **??**) and select the maximum probability per time-step; this results in the path 'SSSRRS' shown in green with the red-arrow deviation. The resulting probability of this approach is found to be 0.0009. However, when you calculate this path (as was done above in equation 6 for a simpler case) the probability of this path is actually Y. What went wrong?

In this example the history of each point was ignored. This was because a step-wise comparison of hidden states was carried out, where the state with the maximum probability is selected, ignoring the history. This is exactly what went wrong in the video.

The Viterbi algorithm solves for this by storing the history of the most probable paths at each step. Below I briefly summarise the Viterbi algorithm as well as provide an alternative possible recursive solution, so read on. *As a disclaimer, I declare that the purpose of the algorithm is not efficiency, but rather a bit of brain teasing that recursion can bring.*

3.5 The Viterbi algorithm

The basic premise of the Viterbi algorithm is to find the maximum likelihood path while avoiding having to carry out all the different hidden state combinations. This is done in a step-wise forward way; first the initial probability (technically only the numerator) is calculated, and for each new time step, the maximum new likelihood is stored together with its history. Finally, the maximum likelihood at the final time step is kept, and its history is backtracked giving rise to the most probable hidden states path.

The following pseudo-code (taken from wiki) summarises this. It uses two arrays T_1 and T_2 to store the maximum probability per step per state and its history respectively. It takes as input the

following:

- the observation space $O = \{o_1, o_2, \ldots, o_N\}$
- the state space $S = \{s_1, s_2, \ldots, s_K\}$
- prior probabilities $\Pi = (\pi_1, \pi_2, \ldots, \pi_K)$ such that π_i stores the probability that $x_1 = s_i$
- a sequence of observations $Y = (y_1, y_2, \ldots, y_T)$ such that $y_t = i$ if the observation at time t is o_i
- transition matrix A of size $K \times K$ such that A_{ij} stores the transition probability of transiting from state s_i to state s_j,
- emission matrix B of size $K \times N$ such that B_{ij} stores the probability of observing o_j from state s_i

and outputs the most likely hidden state sequence $X = (x_1, x_2, \ldots, x_T)$.

```
function VITERBI(O, S, Π, Y, A, B) : X
    for each state i = 1, 2, ..., K do
        T_1[i, 1] ← π_i · B_{i y_1}
        T_2[i, 1] ← 0
    end for

    for each observation j = 2, 3, ..., T do
        for each state i = 1, 2, ..., K do
            T_1[i, j] ← max_k (T_1[k, j − 1] · A_{ki} · B_{i y_j})
            T_2[i, j] ← arg max_k (T_1[k, j − 1] · A_{ki} · B_{i y_j})
        end for
    end for

    z_T ← arg max_k (T_1[k, T])
    x_T ← s_{z_T}
    x_T ← s_{z_T}
    for  j = T, T − 1, ..., 2 do
        z_{j−1} ← T_2[z_j, j]
        x_{j−1} ← s_{z_{j−1}}
        x_{j−1} ← s_{z_{j−1}}
    end for

    return X
end function
```

Listing 1: Python pseudo-code for Viterbi algorithm (taken from wikipedia)

4 Fun recursive solution

The algorithm takes advantage of the recursive nature of the problem. The basic premise is exactly the same as that achieved using the Viterbi algorithm, however the problem is tackled backwards. Instead of starting at time step 2 and and asking the question 'which is the most likely state and where does it come from', it starts at the end and asks 'which previous state returns the maximum probability for the current state' in a recursive fashion. Figure 9 summarises this; the computations start at the leaves of the tree and numbers propagate upwards.

It takes as inputs the set of states, observations, transition and emission probabilities and outputs the most probable hidden states and the probability (just the numerator actually for now). The flow chart algorithm is summarised in **??**. See hmm toy model for code.

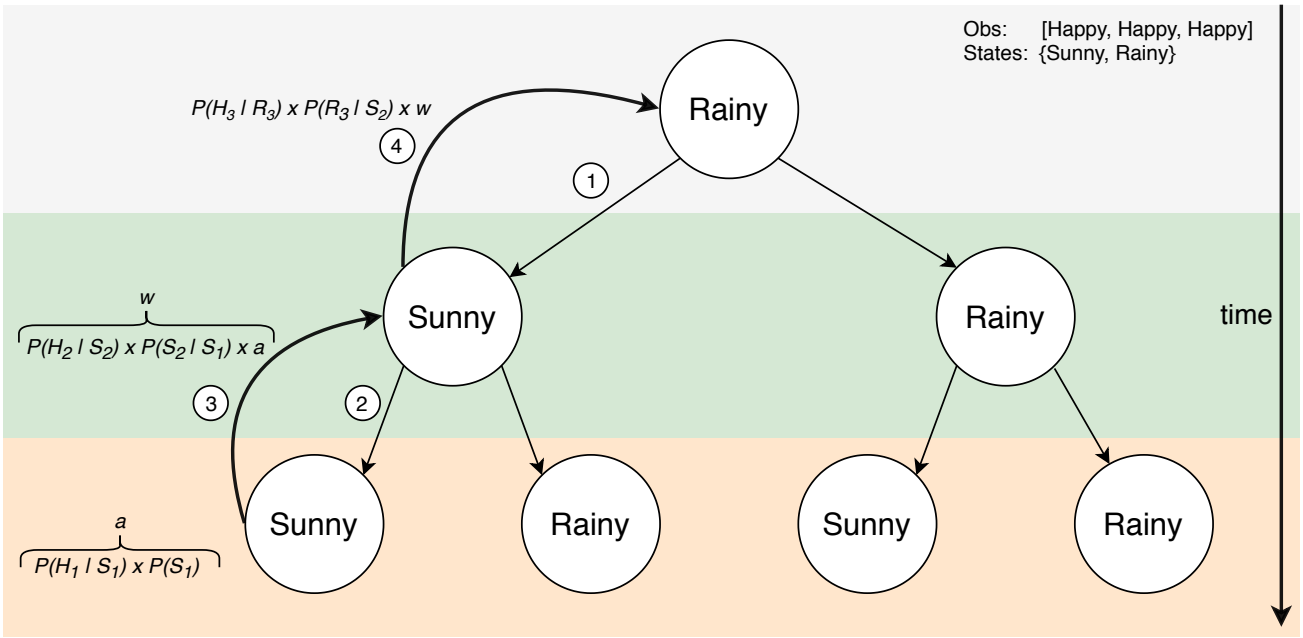

Figure 8: Recursive solution to getting the most probable hidden states.

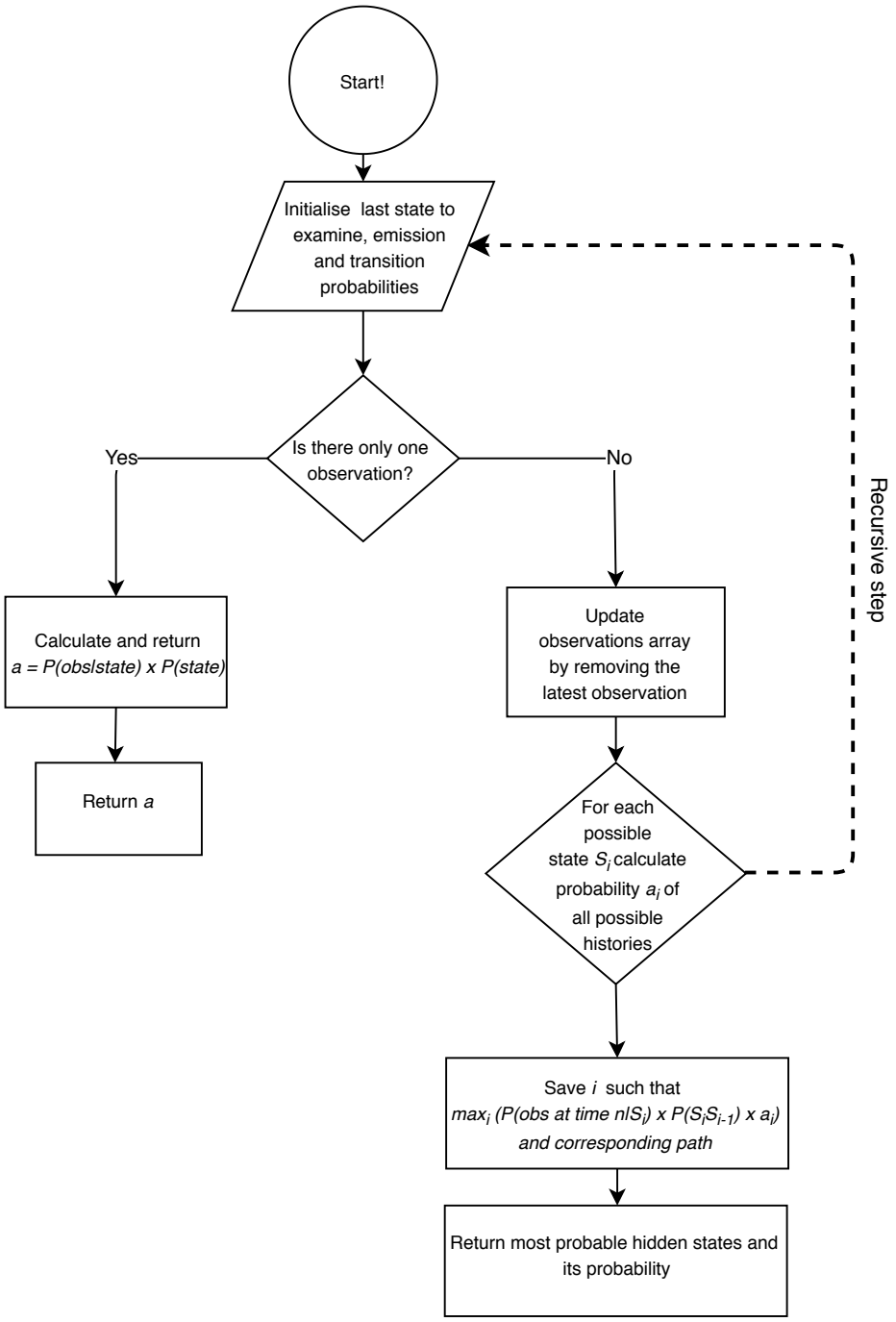

Figure 9: Recursive solution to getting the most probable hidden states.